Networking and Graphs

Prof. Rishman Kaur Jot Chahal

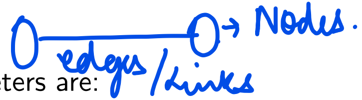
Department of Humanities and Social Sciences
Mehta Family School of Data Science and Artificial Intelligence
Indian Institute of Technology Roorkee

Introduction

- Network science offers a language through which different disciplines can seamlessly interact with each other.
- Several key concepts of network science have their roots in graph theory, a fertile field of mathematics.
- What distinguishes network science from graph theory is its empirical nature i.e. its focus on data, function and utility.
- Today, we gonna learn how to represent a network as a graph and introduce the elementary characteristics of network including degree to degree distribution, from paths to distances, and learn to distinguish weighted and directed network.

Networks and Graphs

- A network is a catalog of a system's components- often called *nodes* or *vertices*- and the direct interactions between them, called *links* or *edges*.



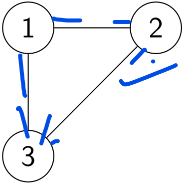
- Two basic network parameters are:
  - The *number of nodes*, or N, represents the number of components in the system. We will often call N *the size of the network*. To distinguish the nodes, we label them i=1,2,3,...,N.

  - The *number of links*, which we denote by L represents the total number of interactions between the nodes.

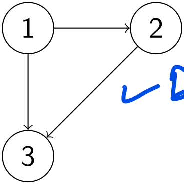
- The links of a network can be *directed* or *undirected*. Some systems have directed links, like the WWW, whose URLs point from one web documents to another or phone calls where one person calls another.

- Other systems have *undirected* links, like romantic ties: if I date Emma Watson, Emma also dates me, or like transmission lines on the power grid, where electric current can flow in both directions.
- Thus, a network is called directed(or a digraph) if all its links are directed; it is called undirected if all its links are undirected. Some networks simultaneously have directed and undirected links.

- The Network shown in figure below includes N=3, L=3. and it is known as undirected graph and directed graph respectively.

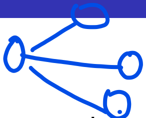


Undirected $K_1 = 2, \; K_2 = 2, \; K_3 = 2.$



Directed.

Degree, Average Degree and Degree Distribution



- A key property of each node is its *degree*, representing the number of links it has to others.

- Degree: We denote by k the degree of the ith node in the network. for example, for undirected networks shown in the above figure, we have $k_1 = 2$, $k_2 = 2$, $k_3 = 2$
- In an undirected network *the total number of links*, L can be expressed as the sum of the node degrees:

$$L = 1/2 \sum_{i=1}^{N} k_i$$

- Here the factor 1/2 corrects for the fact that in the sum above each link is counted twice. For example, the link connecting nodes 1 and 3 will be counted once in the degree of node 1 and once in the degree of node 3.

- Average Degree: An important property of a network is its *average degree*, which for undirected network is

$$< k >= \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2L}{N}$$

.

- In directed networks we distinguish between the *incoming degree*, k_i^{in} representing the number of links that points to node i and the *outgoing degree*, k_i^{out} representing the number of links that point from node i to other nodes. finally, a node's total degree, k_i, is given by

$$k_i = k_i^{in} + k_i^{out}$$

- For example, on the WWW the number of pages a given document points to represents its outgoing degree, k_i^{out} and the number of documents that point to it represents its incoming degree, k_i. in

- The total number of links in a directed network is

$$L = \sum_{i=1}^{N} k_i^{in} = \sum_{i=1}^{N} k_i^{out}$$

- The two sums separately count the outgoing and the incoming degrees. The average degree of a directed network is

$$< k^{in} > = \frac{1}{N}\sum_{i=1}^{N} k_i^{in} = < k^{out} > = \frac{1}{N}\sum_{i=1}^{N} k_i^{out} = \frac{L}{N}$$

- The *degree distribution*, p_k, provides the probability that a randomly selected node in the network has degree k. Since p_k is a probability, it must be normalised:

$$\boxed{\sum_{k=1}^{\infty} p_k = 1}$$

$P_1, P_2, P_3, \ldots$

$\sum P_k = 1.$

- For a network with N nodes, the degree distribution is the normalised histogram given by

$$p_k = N_k / N$$

- where N_k is the number of degree-k nodes. Hence, the number of degree-k nodes can be obtained from the degree distribution as $N_k = Np_k$.

Building up a link b/w Networks & time series forecasting.

- Classical models based on time series forecasting are much difficult to implement compared to the supervised and unsupervised learning models.

(2,2) y_{t-1}, y_{t-2}.

- For instance, ARIMA gives more importance to immediate data points in the test set and tries to perform well for them but as we get far we see a larger variance in the predicted output.

$+f(y_{t-3}, \dots)$

- Also, these techniques are based on the notion that existing patterns in the time series will continue in the future as well.

- Due to the dynamic nature of the time series data often these assumptions are not met when we say there is non-linear co-relation between the past and the current data points.
- In such scenarios it's better to go with Deep Learning Techniques.
- The 3 main limitations to classical models that can be overcome by deep learning methods, are:
  - Missing values are not supported.
  - Assumption that the data has linear relationships.
  - These models work on uni-variate data. Most of the models in time series forecasting don't support multiple variables to be taken as inputs.

Random Networks

- Imagine organising a party for 100 guests who initially do not each other. Once you offer them wine and cheese you will see that soon they start chatting in groups of two or three.
- Now, to Mary one of your guests you mention that red wine in an unlabeled dark green bottle is a vintage and better than other ones. If she shares this information only with her acquaintances your expensive wine appears to be safe (so far).

M

A, B, C

- However, the guests will continue to mingle creating a subtle paths between individuals who may still be stranger to each other.
- Remember in networks physical distance is replaced by *path length*. A *path* is a route that runs along the links of the network. A path's length represents number of links the path contains. (Shortest path is one with fewest number of links to connect same nodes.)

- Now, in the party there is another guest John who did not meet Mary but both of them met Mike. So there is an invisible path from John to Mary.
- As the time goes on, guests will keep on mingling with each other and that secret of vintage wine bottle will pass from Mary to Mike and from Mike to John and so on.
- This is the property of a real network. Most networks that we see do not have a predictable architecture of a web. At first they look very random.
- So, random network theory embraces the apparent randomness by constructing and characterizing networks that are truly random.
- Thus a random network consists of N nodes where each node pair is connected with probability p.

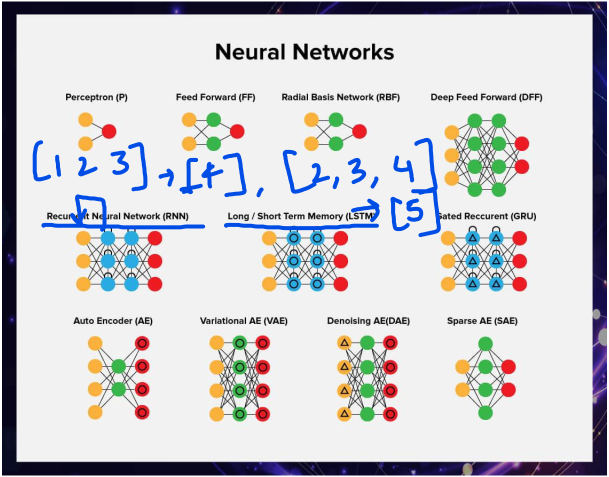
Neural Networks

- Artificial Neural Networks (ANN) is a computational model that consists of several processing elements that receive inputs and deliver outputs based on their predefined activation functions.
- Learns from huge volume of data and uses complex algorithms to train a neural networks.
- Lets make it a bit simpler. Do you ever wonder how Google's auto complete feature predicts the rest of words a user is typing?

Dear Sir, .

- It analyses the data by finding the sequence of words occurring frequently and builds a model to predict the next word in the sentence.
- Neural networks in deep learning, consists of different layers connected to each other and work on the structure and functions of a human brain.

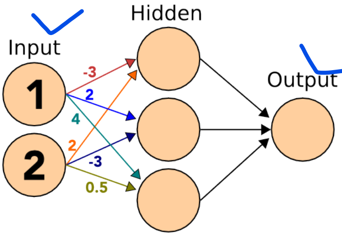
Some types of Neural Networks



- In FF, input moves in one direction.
- Input - Hidden layers - Output layers

- So, its a process where the inputs are propogated through the neural network starting from the input layer through the hidden layers and finally reaches the output layer.

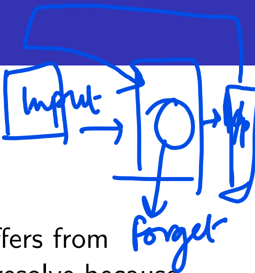
- It is used to make predictions or generate output of a neural network.



Ex (1) Mary buy Mangos. F. PL.

(2) John buys Appl. PL.

Ex M



- However, input data in the time series often suffers from sequence dependence problem which RNN can resolve because they are adaptive in nature.

- and other variants such as $\boxed{\text{LSTM}}$ (Long Short Term Memory) and GRU (Gated Recurrent Units) which are easier to be trained on large "hidden" architecture and achieve better results.

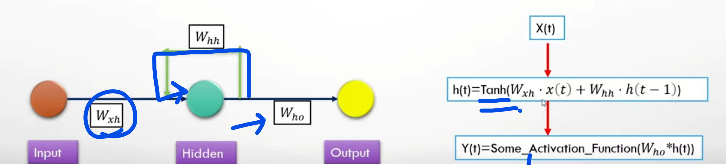
Why RNN?

- What is the problem with ANN that made us shift to RNN?
  - Usually ANN fails where you need history and past sequence of data.
  - In which scenarios it happens?
    - Time Series Forecasting
    - Stock Trade Prediction
    - Natural Language Processing
    - Transportation Crowding predicting problems

Why RNN?

- Let us look at a NLP Task.
  - training _
  - This call may be recorded for training _ purposes.

    history .
- How RNN works? It uses history and introduces a loop.



tanh(-1,1). sigmoid (0,1)

Why RNN?

- If history is so important why can't we feed it to an ANN?
  - Can use bag of word approach where each word will be assigned a number and we can use that for prediction.
  - This is a plausible approach. Then, what is the issue?

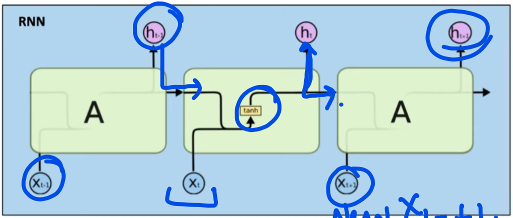
sequence

    - Order of words - which word came first and which came second.
  - **Sentiment Analysis Example:** He is not a great cook, but instead very bad. (positive or negative review). Now, here sequence is important.
    - He is not a great cook, but instead really bad. (Negative review)
    - He is not a bad cook, but instead really great. (Positive review)
  - So, RNN keeps the order of the sequence intact.

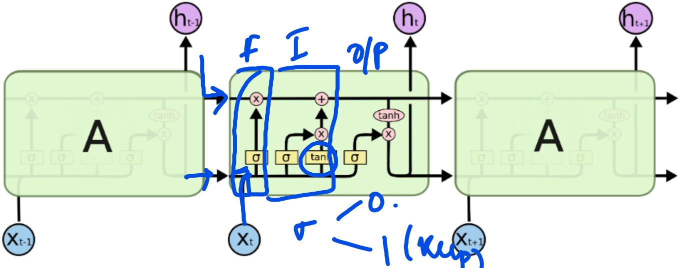
Long short term memory networks (LSTM)

- LSTMs are a special kind of recurrent networks, capable of learning long term dependencies.
- Or in other words, they are capable of remembering information for long periods of time.
- All recurrent neural networks have the form of a chain of repeating modules of neural networks, such as single tanh module.
- But LSTM model has a combination of four layers interacting with each other instead of having a single neural network.

More Formally: RNN and LSTM



New x_t. New x_{t+1}.



The repeating module in an LSTM contains four interacting layers.

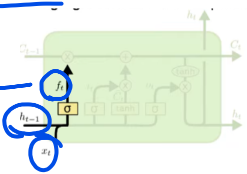
Working of LSTMs

- The input gate controls the information flow to the current cell state using a point-wise multiplication operation of 'sigmoid' and 'tanh' respectively.

- Finally, the output gate decides which information should be passed on to the next hidden state.

- **STEP 1**: It decides which information to be omitted in from the cell in that particular time step. It is decided by the sigmoid function.
  $f_t = \sigma(W_f.[h_t-_1, x_t] + b_f$
- It computes the forget gate function (f_t) by looking at the previous state (h_t-_1) and the current input (x_t)

forget

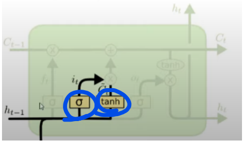


Cont..

- **STEP 2** Decides how much should this unit adds to the current state. This layer consists of 2 functions that is sigmoid and tanh.

- $i_t = \sigma(W_i.[h_t{-}_1, x_t] + b_i$
- $C_t = tanh(W_c.[h_t{-}_1, x_t] + b_c)$

- Sigmoid function decides which value to let through and tanh function gives the weightage to the values which are passed deciding their level of importance.



Cont..

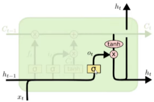
- **STEP 3** Decides what part of the current cell state makes it to the output or to decide what will be our output.

- $o_t = \sigma(W_o.[h_t - 1, x_t] + b_o$
- $h_t = O_t * tanh(C_t)$

- o_t is the output gate.

- First, we run a sigmoid layer to which decides what parts of the cell state makes it to the output.

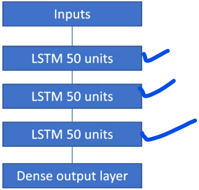
- Then we put the cell state through tanh to push the values to be between -1 and 1 and multiply it by the output of the sigmoid function.



Cont..

- LSTM in keras

```
model = Sequential()

model.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
model.add(Dropout(0.2))

model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))

model.add(LSTM(units = 50))
model.add(Dropout(0.2))

model.add(Dense(units = 1))

model.compile(optimizer = 'adam', loss = 'mean_squared_error')

model.fit(X_train, y_train, epochs = 100, batch_size = 32)
```



Thanking You